Assignment 5: Uncompressing files

ECED 3401 - Systems Analysis

Department of Electrical and Computer Engineering
Dalhousie University

Background

Data compression is used by companies when the volume of data being sent or stored needs to be reduced in size. For example, a streaming service such as Netflix sends compressed files from its servers to its customers, where the customer's Netflix application software *uncompresses* the file for viewing.

A simple scheme for creating a compressed version of a text file can be used for files which contain no digit characters.

The *compression* scheme requires making a list of the words in the uncompressed file.

- When a non-alphabetic character (e.g., punctuation) is encountered in the uncompressed file, it is copied directly into the compressed file.
- When a word (any alphabetic character or characters) is encountered in the uncompressed file, it is copied directly into the compressed file only if this is the first occurrence of the word. In this case, the word is put at the front of the list.
- However, if the word is not its first occurrence, the word is not copied to the compressed file. Instead, its position in the list is copied into the compressed file and *then* the word is moved to the front of the list.

The numbering of list positions begins at 1.

Requirements

You are to design, implement, and test a program that takes a *compressed* file as input and generates a reproduction of the original *uncompressed* file as output, either to the screen or a file. You can assume that no word contains more than 50 characters and that the *original* uncompressed file contains no digit characters.

For the purposes of this problem, a word is defined to be a sequence of upper- and lower-case letters. Words are case-sensitive, the word "abc" is not the same as the word "Abc". For example,

x-ray contains 2 words: "x" and "ray"
Sandy's contains 2 words: "Sandy" and "s"
It's a winner contains 4 words: "It", "s", "a", and "winner"

There is no upper limit on the number of different words in the input file. The end of the input file is signified by the number '0' on a line by itself. The terminating '0' merely indicates the end of the input and should not be part of the output produced by your program. If the terminal '0' is missing, the program should stop if end-of-file is encountered.

The clever thing about this algorithm is that a dictionary of words is not required. The most recently used words are kept at or near the front of the list. The uncompressed file can be extracted from the information supplied in the compressed file.

Sample Input (Compressed file)

```
One Fish Two 2 Red 2 Blue 2
Dr. Seuss

7 fish, 8 2, 8 2, 8 2,
Black 2, 3 2, Old 2, New 2.
This one has a little car.
6 6 6 6 6 star.
Say! What 5 lot of 12 there are.
Yes. Some 3 red, and some 4 blue.
6 3 old 6 6 4 new.
6 3 sad, 6 6 4 glad,
And 4 4 very, 1 bad.
Why 4 they 10 10 10 2 7?
0
```

Output for the Sample Input (Uncompressed file)

```
One Fish Two Fish Red Fish Blue Fish
Dr. Seuss

One fish, Two fish, Red fish, Blue fish,
Black fish, Blue fish, Old fish, New fish.
This one has a little car.
This one has a little star.
Say! What a lot of fish there are.
Yes. Some are red, and some are blue.
Some are old and some are new.
Some are sad, and some are glad,
And some are very, very bad.
Why are they sad and glad and bad?
```

You can write your output to the screen (for a screenshot) or to a file.

Submission

You are to submit a design document, the source code of your solution, and a test report containing your test results.

Marking

The assignment will be marked as follows:

Design

A detailed structured-English (pseudocode) description of your solution.

The design should be sufficiently detailed to allow a C programmer (other than yourself) to implement the solution.

Total points: 2.

Software

A fully commented, properly indented, magic-numberless, tidy piece of modular software that meets the requirements described above and meets the design description. Avoid “hard-coding” solutions into the application. The solution should be “general purpose”, allowing the programmer to modify the system by changing the data, not the software.

Total points: 5

Testing

You are to develop your own set of at least four tests that demonstrate how your software works in different conditions. The testing format used in Assignment 1 is to be used in this assignment. Input files, rather than standard input, are to be used. Your tests must contain files other than those shown in this assignment.

Total points: 3.

Dates

Available: 6 October 2022 (6:00pm)

Due: 16 October 2022 (11:59pm)

Notes

Uncompressing the file uses the compression algorithm to create the list. The difference being, when a number is read its position in the list is found, the word is displayed, and the word is moved to the head of the list.

A compression executable can be found on the course Teams site (General > Class Materials > Compress.exe). The executable lets you generate your own compressed files for testing. If you study these, you will understand how uncompressing can be implemented.

All data should be read as character, which means numbers will need to be converted to integer.

If you decide to write to a file, `fprintf()` can be used for output.

This assignment is worth 3% of your overall grade.

Late assignments will be penalized one point per day starting on 17 October 2022 at midnight (00:00am).

Late assignments will not be accepted once a solution has been discussed in class.

This is the final assignment of the term.

You must work on this assignment by yourself.

If you are having difficulties with this assignment or any part of the course, please contact Dr. Hughes or one of the course TAs.